CLAIMS

What is claimed is:

1. A remote file system, comprising:

one or more surrogate providers comprising at least a first surrogate provider that selectively caches at least a subset of data from at least one online server; and

one or more client computers that receive and store the subset of data to their respective local databases for offline use by the respective client computers to facilitate a seamless operation of data retrieval across connectivity states for a user.

2. The system of claim 1, the first surrogate provider is a client side caching (CSC) component that supports connection state transitions at the directory level on a logical namespace.

3. The system of claim 1, further comprising an MUP that supports the one or more surrogate providers at the directory level to handle incoming requests from a user.

4. The system of claim 1, further comprising a second surrogate provider that translates a logical path into a physical path.

5. The system of claim 4, the second surrogate provider is a DFS component that points to at least one physical share or at least one physical server.

6. The system of claim 1, wherein selectively caching comprises automatic caching and manual caching based at least in part upon user preferences.

7. The system of claim 1, the data comprises file access parameters comprising at least one of object access rights and share access rights, the file access parameters corresponding to a cached file object.

8. The system of claim 2, the CSC component caches the logical namespace of a file request such that when accessed during an offline state, the file is presented to a user as if it resides at a remote server location.

9. The system of claim 2, the CSC component maintains connection based data structures in logical namespace, the data structures comprising a server connection structure (SrvCall), a share mapping structure (NetRoot), and a per-user share mapping structure (VNetRoot) to facilitate handling at least one of create, read, and write requests.

10. The system of claim 2, the CSC component creates file based data structures and shares the data structures with one or more redirectors to facilitate handling at least one of create, read, and write requests, the one or more redirectors operatively connected to one or more network providers.

11. The system of claim 1, the first surrogate provider comprises a pre-process handler and a post-process handler which facilitates responding to any one of create, read, and write requests.

12. The system of claim 2, the surrogate providers determine who owns a path request whereby the CSC components makes an initial determination before allowing the DFS component to examine the path to identify any DFS links.

13. The system of claim 12, the CSC component operates cooperatively with the DFS component to determine whether DFS links are present in the path while in an online connection state.

14. The system of claim 2, the CSC component determines whether to cache an object file associated with the path.

15. The system of claim 2, further comprising a CSC agent pings the server to determine whether the server is online.

16. The system of claim 2, the CSC component tracking substantially all DFS links included in the logical namespace persistently to transition a connection state at a proper logical directory which facilitates minimizing a scope of offlineness to a physical share.

17. The system of claim 1, the server broadcasts to substantially all CSC agents that it is online to mitigate latency.

18. The system of claim 1, the client computer accesses remote files offline by retrieving them from their respective local databases if file access parameters are satisfied.

19. The system of claim 1, the first surrogate provider keeps track of DFS links corresponding to every object, wherein the DFS links are physical shares.

20. The system of claim 1, the first surrogate provider determines whether the request against a specific object should be carried out offline or not, before returning to MUP, by looking at a corresponding physical share connection state.

21. A method that facilitates maintaining access to remote files (*e.g.,* server-based) during any period of disconnect from a remote location, comprising:

providing one or more client computers, each client computer comprising a local data store; and

selectively caching one or more file objects from at least one online server to the respective data store for subsequent offline use by the client computer.

22. The method of claim 21, further comprising maintaining access to the one or more files cached while offline.

23. The method of claim 21, further comprising caching one of more file access parameters that correspond to the one or more cached file objects to permit client access to the file objects while offline.

24. The method of claim 21, when connected to the remote location, retrieving a file object from the local data store to mitigate bandwidth usage with respect to accessing the remote location despite being connected to the remote location.

25. The method of claim 21, further comprising:

mapping a logical namespace to a physical namespace to facilitate keeping track of cached files and enumerating directories as files are modified or deleted locally at the client or at the remote location; and

tracking connection states and version of physical shares that correspond to at least one object along a path that facilitates updating a tree connect structure in a continuous manner.

26. A method that facilitates seamless operation across connectivity states between at least one client and at least one remote server, comprising:

providing at least a first surrogate provider that receives one or more I/O requests from an MUP, the first surrogate provider comprising a pre-process handler and a post-process handler that facilitate handling the requests at a directory level, the first surrogate provider examining a logical path of the request; and

passing the one or more requests to a second surrogate provider that is operational in an online state, the second surrogate provider translating the logical path of the request into a physical path; and

generating one or more data structures for each respective I/O request that facilitates determining whether the first surrogate provider wants to own or cache a file object related to the request.

27. The method of claim 26, further comprising:

processing the request using the pre-process handler to determine whether the request was handled by at least one of a network provider and the first surrogate provider;

optionally calling the post-process handler after the request is handled to handle the request again;

optionally passing the request to a second surrogate provider, the second surrogate provider examines the request and maps the request path to a physical path at the directory level; and

optionally passing the request to one or more redirectors to allow the one or more redirectors to claim ownership of a file object requested.

28. The method of claim 26, the request is a create request.

29. The method of claim 26, the first surrogate provider is a CSC component and the second surrogate provider is a DFS component, the DFS identifying DFS links in cooperation with the CSC component only while online.

30. The method of claim 26, the request being one of a read and a write operation request.

31. The method of claim 30, the first surrogate provider is provided with a buffering state of a file before substantially every read from a persistent cache to a client application or before substantially every write is executed, respectively.

32. The method of claim 26, employing the first surrogate provider to keep track of DFS links corresponding to every object, wherein the DFS links are physical shares.

33. The method of claim 26, employing the first surrogate provider to determine whether the request against a specific object should be carried out offline or not, before returning to MUP, by looking at a corresponding physical share connection state.

34. An API that facilitates satisfying a create request on an online remote file system comprising:

receive the create request from I/O manager;

call a pre-process handler of a CSC surrogate provider;

find or create a logical namespace structure if part of the logical namespace on which a target of the create request resides is already offline;

pass the create request to a DFS surrogate provider to translate the logical path to an physical server share;

pass the create request to a redirector component to allow a redirector to claim the physical path; and

call a post-process handler of the CSC surrogate provider to express one of either no interest or interest to cache a file object requested by the create request.

35. An API that facilitates satisfying a create request on a client computer when disconnected from a remote file system comprising:

receive the create request from I/O manager; and

call a pre-process handler of a CSC surrogate provider to handle the request by mapping the logical path to local cache data since redirectors are unavailable to claim the path.

36. A system that facilitates maintaining access to remote files (*e.g.,* server-based) during any period of disconnect from a remote location, comprising:

means for providing one or more client computers, each client computer comprising a local data store; and

means for selectively caching one or more file objects from at least one online server to the respective data store for subsequent offline use by the client computer.

37. The system of claim 36, further comprising means for maintaining access to the one or more files cached while offline.

38. The system of claim 36, further comprising means for caching one of more file access parameters that correspond to the one or more cached file objects to permit client access to the file objects while offline.

39. The system of claim 36, when connected to the remote location, means for retrieving a file object from the local data store to mitigate bandwidth usage with respect to accessing the remote location despite being connected to the remote location.

40. The system of claim 36, further comprising:

means for mapping a logical namespace to a physical namespace to facilitate keeping track of cached files and enumerating directories as files are modified or deleted locally at the client or at the remote location; and

means for tracking connection states and version of physical shares that correspond to at least one object along a path that facilitates updating a tree connect structure in a continuous manner.

41. A system that facilitates seamless operation across connectivity states between at least one client and at least one remote server, comprising:

means for providing at least a first surrogate provider that receives one or more I/O requests from an MUP, the first surrogate provider comprising a pre-process handler and a post-process handler that facilitate handling the requests at a directory level, the first surrogate provider examining a logical path of the request; and

means for passing the one or more requests to a second surrogate provider that is operational in an online state, the second surrogate provider translating the logical path of the request into a physical path; and

means for generating one or more data structures for each respective I/O request that facilitates determining whether the first surrogate provider wants to own or cache a file object related to the request.

42. A data packet adapted to be transmitted between two or more computer processes facilitating extracting data from messages, the data packet comprising:

information associated with providing one or more client computers, each client computer comprising a local data store, selectively caching one or more file objects from at least one online server to the respective data store for subsequent offline use by the caching one of more file access parameters that correspond to the one or more cached file objects to permit client access to the file objects while offline in connection with seamless connection state transitions at a directory level.

43. A computer readable medium storing computer executable components of claim 1.